GP - Abalone

Jieying Ma

2023-12-07

1. statistic analysis

```
abalone = read.csv("abalone/abalone.txt", header = FALSE, col.names = c("Sex","Length","Diameter","Height","Whole
_weight","Shucked_weight","Viscera_weight","Shell_weight","Rings"))

# install.packages("psych")
library(psych)
describe(abalone)
```

	vars <int>	n <dbl>	mean <dbl>	sd <dbl>	median <dbl>	trimmed <dbl>	mad <dbl>	min <dbl>	max <dbl>
Sex*	1	4177	2.0529088	0.82224042	2.0000	2.0661083	1.4826000	1.0000	3.0000
Length	2	4177	0.5239921	0.12009291	0.5450	0.5324783	0.1186080	0.0750	0.8150
Diameter	3	4177	0.4078813	0.09923987	0.4250	0.4146994	0.0963690	0.0550	0.6500
Height	4	4177	0.1395164	0.04182706	0.1400	0.1402498	0.0370650	0.0000	1.1300
Whole_weight	5	4177	0.8287422	0.49038902	0.7995	0.7995646	0.5285469	0.0020	2.8255
Shucked_weight	6	4177	0.3593675	0.22196295	0.3360	0.3439231	0.2349921	0.0010	1.4880
Viscera_weight	7	4177	0.1805936	0.10961425	0.1710	0.1733193	0.1178667	0.0005	0.7600
Shell_weight	8	4177	0.2388309	0.13920267	0.2340	0.2305173	0.1475187	0.0015	1.0050
Rings	9	4177	9.9336845	3.22416903	9.0000	9.6410410	2.9652000	1.0000	29.0000

9 rows | 1-10 of 14 columns

1.1 distribution

```
colnames(abalone)
```

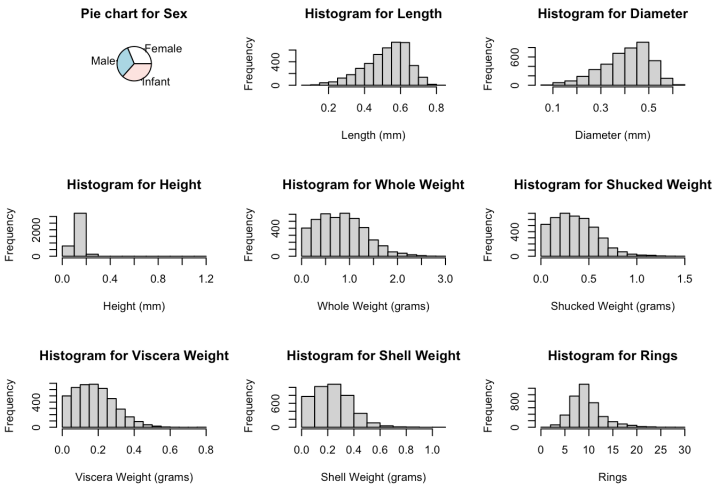
```
## [1] "Sex"            "Length"         "Diameter"       "Height"
## [5] "Whole_weight"   "Shucked_weight" "Viscera_weight" "Shell_weight"
## [9] "Rings"
```

```
abalone$Sex = as.factor(abalone$Sex)
sapply(abalone, typeof)
```

```
##            Sex         Length       Diameter         Height   Whole_weight
##      "integer"       "double"       "double"       "double"       "double"
## Shucked_weight Viscera_weight   Shell_weight          Rings
##       "double"       "double"       "double"      "integer"
```

```
library(vioplot)
```

```
par(mfrow=c(3,3))
pie(table(abalone$Sex), labels = c('Female','Male','Infant'), main="Pie chart for Sex")
hist(abalone$Length,main="Histogram for Length", xlab="Length (mm)")
hist(abalone$Diameter,main="Histogram for Diameter", xlab="Diameter (mm)")
hist(abalone$Height,main="Histogram for Height", xlab="Height (mm)")
hist(abalone$Whole_weight,main="Histogram for Whole Weight", xlab="Whole Weight (grams)")
hist(abalone$Shucked_weight,main="Histogram for Shucked Weight", xlab="Shucked Weight (grams)")
hist(abalone$Viscera_weight,main="Histogram for Viscera Weight", xlab="Viscera Weight (grams)")
hist(abalone$Shell_weight,main="Histogram for Shell Weight", xlab="Shell Weight (grams)")
hist(abalone$Rings,main="Histogram for Rings", xlab="Rings")
```



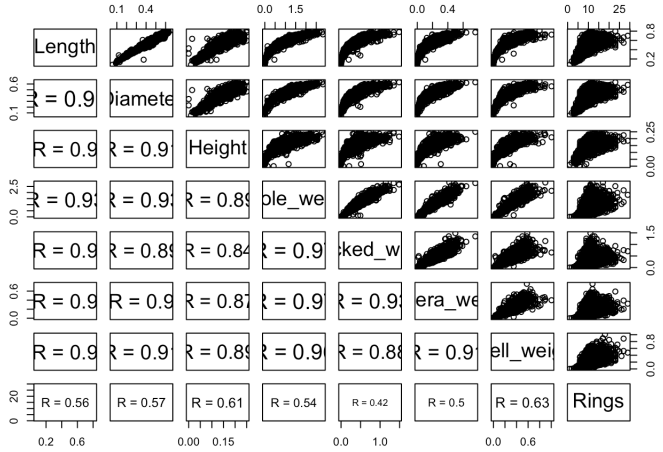

```
par(mfrow=c(1,1))
```

```
abalone1 = abalone
max_heights = tail(sort(abalone$Height), 2)
indexes_to_remove = which(abalone$Height %in% max_heights)
abalone = abalone[-indexes_to_remove, ]
```

1.2 correlation

```
panel.cor = function(x, y){
  par(usr = c(0, 1, 0, 1))
  r = round(cor(x, y, use="complete.obs"), 2)
  txt = paste0("R = ", r)
  text(0.5, 0.5, txt, cex = 2 * r)
}

pairs(~Length+Diameter+Height+Whole_weight+Shucked_weight+Viscera_weight+Shell_weight+Rings,data=abalone, lower.panel = panel.cor, cex.labels = 1.8)
```




```
describe(abalone)
```

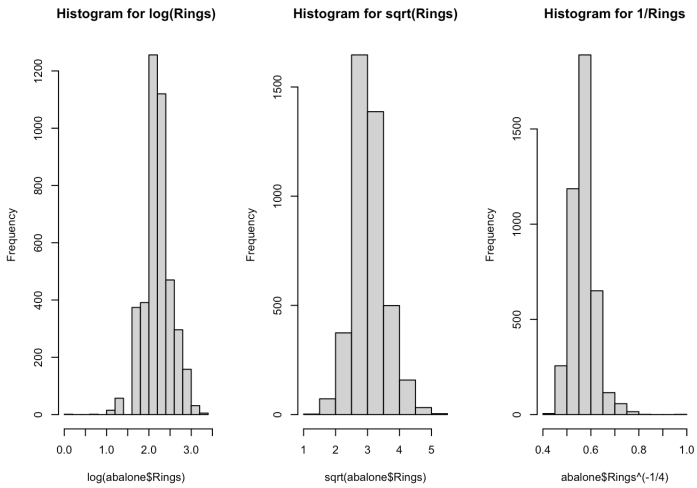
	vars	n	mean	sd	median	trimmed	mad	min	max
	<int>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
Sex*	1	4175	2.0529341	0.82214521	2.0000	2.0661479	1.4826000	1.0000	3.0000
Length	2	4175	0.5239653	0.12008425	0.5450	0.5324633	0.1186080	0.0750	0.8150
Diameter	3	4175	0.4078563	0.09923046	0.4250	0.4146842	0.0963690	0.0550	0.6500
Height	4	4175	0.1391892	0.03848917	0.1400	0.1402230	0.0370650	0.0000	0.2500
Whole_weight	5	4175	0.8284675	0.49002679	0.7995	0.7994229	0.5285469	0.0020	2.8255
Shucked_weight	6	4175	0.3591949	0.22171345	0.3360	0.3438360	0.2349921	0.0010	1.4880
Viscera_weight	7	4175	0.1805358	0.10953364	0.1710	0.1732908	0.1178667	0.0005	0.7600
Shell_weight	8	4175	0.2387907	0.13916225	0.2340	0.2304897	0.1475187	0.0015	1.0050
Rings	9	4175	9.9341317	3.22480229	9.0000	9.6414247	2.9652000	1.0000	29.0000

9 rows | 1-10 of 14 columns

```
# install.packages("openxlsx")
library(openxlsx)
```

```
# write.xlsx(describe(abalone), "describe_output.xlsx")
```

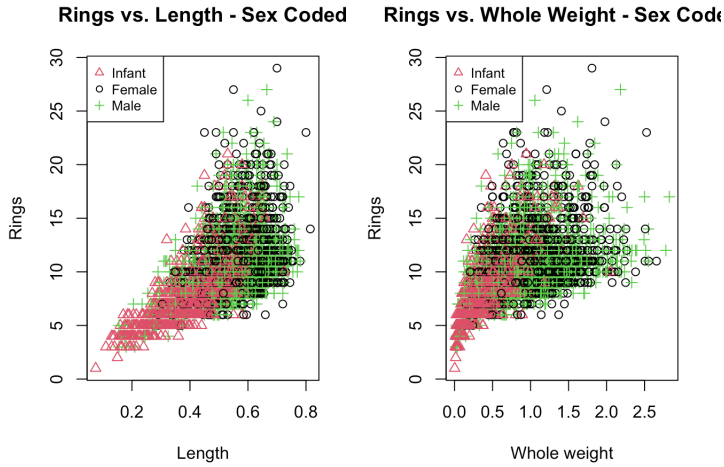
```
par(mfrow=c(1,3))
hist(log(abalone$Rings),main="Histogram for log(Rings)")
hist(sqrt(abalone$Rings),main="Histogram for sqrt(Rings)")
hist(abalone$Rings^(-1/4),main="Histogram for 1/Rings")
```



1.3 Sex Coded

```
par(mfrow=c(1,2))
plot(abalone$Length,abalone$Rings,pch=as.integer(abalone$Sex),
      col=as.integer(abalone$Sex),main='Rings vs. Length - Sex Coded',
        ylab='Rings',xlab='Length')
legend('topleft',legend=c('Infant','Female','Male'),
        pch=c(2,1,3),col=c(2,1,3), cex = 0.8) # add a legend

plot(abalone$Whole_weight,abalone$Rings,pch=as.integer(abalone$Sex),
      col=as.integer(abalone$Sex),main='Rings vs. Whole Weight - Sex Coded',
        ylab='Rings',xlab='Whole weight')
legend('topleft',legend=c('Infant','Female','Male'),
        pch=c(2,1,3),col=c(2,1,3), cex = 0.8) # add a legend
```




```
## jpeg("output_plot.jpeg", width = 8, height = 6)
par(mfrow=c(1,1))
```

2. Model choose

```
## response variable
y = abalone[,9]+1.5     # rings→years
data = cbind(abalone[,-9],y) # [x1-8,y]
```

```
## split the data
set.seed(100)
n = nrow(data)/3
ind = sample(1:(3*n), n, replace=FALSE)
test = data[ind, ]
train = data[-ind, ]

x1=train[,1]  # sex
x2=train[,2]  # length
x3=train[,3]  # diameter
x4=train[,4]  # height
x5=train[,5]  # whole_weight
x6=train[,6]  # shucked_weight
x7=train[,7]  # viscera_weight
x8=train[,8]  # shell_weight
y=train[,9]   # y - age
```

2.1 AIC&BIC ON FIRST ORDER

```
library(MASS)
none_mod <- lm(y~1, data=train) # model with only intercept
full_mod <- lm(y~., data=train) # first order full model
library(car)
# forward stepwise based on AIC
fit1=stepAIC(none_mod, scope=list(upper=full_mod, lower = ~1), direction="both", k=2, trace = FALSE)
vif(fit1)
```

```
##                     GVIF Df GVIF^(1/(2*Df))
## Shell_weight   20.025209  1        4.474954
## Shucked_weight 25.999625  1        5.098983
## Height          6.618462  1        2.572637
## Whole_weight   97.857701  1        9.892305
## Sex             1.519451  2        1.110252
## Viscera_weight 16.503660  1        4.062470
## Diameter        9.423914  1        3.069839
```

```
# forward selection based on AIC:
fit2=stepAIC(none_mod, scope=list(upper=full_mod, lower = ~1), direction="forward", k=2, trace = FALSE)
vif(fit2)
```

```
##                     GVIF Df GVIF^(1/(2*Df))
## Shell_weight   20.025209  1        4.474954
## Shucked_weight 25.999625  1        5.098983
## Height          6.618462  1        2.572637
## Whole_weight   97.857701  1        9.892305
## Sex             1.519451  2        1.110252
## Viscera_weight 16.503660  1        4.062470
## Diameter        9.423914  1        3.069839
```

```
# backward elimination based on AIC
fit3=stepAIC(full_mod, scope=list(upper=full_mod, lower = ~1), direction="backward", k=2, trace = FALSE)
vif(fit3)
```

```
##                     GVIF Df GVIF^(1/(2*Df))
## Sex             1.519451  2        1.110252
## Diameter        9.423914  1        3.069839
## Height          6.618462  1        2.572637
## Whole_weight   97.857701  1        9.892305
## Shucked_weight 25.999625  1        5.098983
## Viscera_weight 16.503660  1        4.062470
## Shell_weight   20.025209  1        4.474954
```

```
# backward stepwise based on AIC
fit4=stepAIC(full_mod, scope=list(upper=full_mod, lower = ~1), direction="both", k=2, trace = FALSE)
vif(fit4)
```

```
##                     GVIF Df GVIF^(1/(2*Df))
## Sex             1.519451  2        1.110252
## Diameter        9.423914  1        3.069839
## Height          6.618462  1        2.572637
## Whole_weight   97.857701  1        9.892305
## Shucked_weight 25.999625  1        5.098983
## Viscera_weight 16.503660  1        4.062470
## Shell_weight   20.025209  1        4.474954
```

```
model1 = fit1
summary(fit1)
```

```
##
## Call:
## lm(formula = y ~ Shell_weight + Shucked_weight + Height + Whole_weight +
##     Sex + Viscera_weight + Diameter, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.1538 -1.3191 -0.3653  0.8610 14.2845
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)      5.21471    0.34283  15.211  < 2e-16 ***
## Shell_weight     7.50702    1.34261   5.591 2.47e-08 ***
## Shucked_weight -18.81258    0.95799 -19.638  < 2e-16 ***
## Height          23.09499    2.78828   8.283  < 2e-16 ***
## Whole_weight     8.48281    0.83969  10.102  < 2e-16 ***
## SexI            -0.81367    0.12486  -6.517 8.51e-11 ***
## SexM            -0.01247    0.10215  -0.122    0.903
## Viscera_weight -10.00798    1.53332  -6.527 7.95e-11 ***
## Diameter         7.38376    1.29211   5.715 1.22e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.204 on 2775 degrees of freedom
## Multiple R-squared:  0.5292, Adjusted R-squared:  0.5278
## F-statistic: 389.9 on 8 and 2775 DF,  p-value: < 2.2e-16
```

All kinds of AIC give the same model - model 1 in which length is dropped,

Age_{Female} = 5.21471 + 7.50702 * Shell Weight − 18.81258 * Shucked_weight + 23.09499 * Height + 8.48281 * Whole_weight − 10.00798 * Viscera_w

Age_{Male} = 5.20224 − 0.01247 + 7.50702 * Shell Weight − 18.81258 * Shucked_weight + 23.09499 * Height + 8.48281 * Whole_weight − 10.00798 *

Age_{Infant} = 4.40104 + 7.50702 * Shell Weight − 18.81258 * Shucked_weight + 23.09499 * Height + 8.48281 * Whole_weight − 10.00798 * Viscera_w

```
vif_1st = t(vif(fit1))
vif_1st
```

```
##                 Shell_weight Shucked_weight   Height Whole_weight      Sex
## GVIF              20.025209      25.999625 6.618462    97.857701 1.519451
## Df                 1.000000       1.000000 1.000000     1.000000 2.000000
## GVIF^(1/(2*Df))    4.474954       5.098983 2.572637     9.892305 1.110252
##                 Viscera_weight Diameter
## GVIF                 16.50366 9.423914
## Df                    1.00000 1.000000
## GVIF^(1/(2*Df))       4.06247 3.069839
```

The VIF values of Whole_weight is very high(97.86), indicating that there may be strong multicollinearity in it. So we drop Whole_weight and fit a model called model 2.

```
# model 2 without Whole_weight
model2 <- lm(y ~ Shell_weight + Shucked_weight + Height + Sex + Viscera_weight + Diameter, data=train)
summary(model2)
```

```
##
## Call:
## lm(formula = y ~ Shell_weight + Shucked_weight + Height + Sex +
##     Viscera_weight + Diameter, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.9362 -1.3335 -0.3736  0.8374 15.9867
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)      5.04300    0.34858  14.467  < 2e-16 ***
## Shell_weight    17.96951    0.86984  20.658  < 2e-16 ***
## Shucked_weight -10.98655    0.57375 -19.149  < 2e-16 ***
## Height          23.98946    2.83715   8.455  < 2e-16 ***
## SexI            -0.87181    0.12698  -6.866 8.12e-12 ***
## SexM            -0.03068    0.10397  -0.295    0.768
## Viscera_weight  -0.84388    1.25850  -0.671    0.503
## Diameter         7.72494    1.31497   5.875 4.74e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.244 on 2776 degrees of freedom
## Multiple R-squared:  0.5119, Adjusted R-squared:  0.5106
## F-statistic: 415.8 on 7 and 2776 DF,  p-value: < 2.2e-16
```

```
vif_2st = t(vif(model2))
vif_2st
```

```
##                 Shell_weight Shucked_weight   Height      Sex Viscera_weight
## GVIF                8.110040       8.998401 6.611788 1.516185      10.727343
## Df                  1.000000       1.000000 1.000000 2.000000       1.000000
## GVIF^(1/(2*Df))     2.847813       2.999733 2.571340 1.109655       3.275262
##                 Diameter
## GVIF            9.417476
## Df              1.000000
## GVIF^(1/(2*Df)) 3.068791
```

```
n=2785
#forward stepwise based on BIC
fit5=stepAIC(none_mod, scope=list(upper=full_mod, lower = ~1), direction="both", k=log(n), trace = FALSE)
vif(fit5)
```

```
##                    GVIF Df GVIF^(1/(2*Df))
## Shell_weight   20.025209  1        4.474954
## Shucked_weight 25.999625  1        5.098983
## Height          6.618462  1        2.572637
## Whole_weight   97.857701  1        9.892305
## Sex             1.519451  2        1.110252
## Viscera_weight 16.503660  1        4.062470
## Diameter        9.423914  1        3.069839
```

```
#forward selection based on BIC:
fit6=stepAIC(none_mod, scope=list(upper=full_mod, lower = ~1), direction="forward", k=log(n), trace = FALSE)
vif(fit6)
```

```
##                    GVIF Df GVIF^(1/(2*Df))
## Shell_weight   20.025209  1        4.474954
## Shucked_weight 25.999625  1        5.098983
## Height          6.618462  1        2.572637
## Whole_weight   97.857701  1        9.892305
## Sex             1.519451  2        1.110252
## Viscera_weight 16.503660  1        4.062470
## Diameter        9.423914  1        3.069839
```

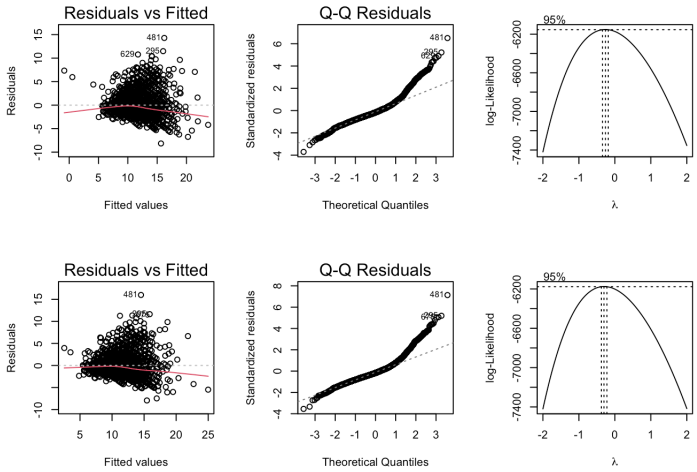
```
#backward elimination based on BIC
fit7=stepAIC(full_mod, scope=list(upper=full_mod, lower = ~1), direction="backward", k=log(n), trace = FALSE)
vif(fit7)
```

```
##                    GVIF Df GVIF^(1/(2*Df))
## Sex             1.519451  2        1.110252
## Diameter        9.423914  1        3.069839
## Height          6.618462  1        2.572637
## Whole_weight   97.857701  1        9.892305
## Shucked_weight 25.999625  1        5.098983
## Viscera_weight 16.503660  1        4.062470
## Shell_weight   20.025209  1        4.474954
```

```
#backward stepwise based on BIC
fit8=stepAIC(full_mod, scope=list(upper=full_mod, lower = ~1), direction="both", k=log(n), trace = FALSE)
vif(fit8)
```

```
##                     GVIF Df GVIF^(1/(2*Df))
## Sex             1.519451  2        1.110252
## Diameter        9.423914  1        3.069839
## Height          6.618462  1        2.572637
## Whole_weight   97.857701  1        9.892305
## Shucked_weight 25.999625  1        5.098983
## Viscera_weight 16.503660  1        4.062470
## Shell_weight   20.025209  1        4.474954
```

```
par(mfrow=c(2,3))
plot(model1, which=1)
plot(model1, which=2)
boxcox(model1)
plot(model2, which=1)
plot(model2, which=2)
boxcox(model2)
```



Residuals vs Fitted (Residuals vs Fitted values plot):

In the residuals plot, there is not a completely random distribution between the residuals and the fitted values, and the center shows some curvilinear trends, indicating that the model may have a nonlinear relationship. This suggests that the model's assumption of linearity may not be completely valid.

The residuals in the center portion are more concentrated, while the distribution of residuals at the extremes is more dispersed, indicating that the assumption of homoscedasticity may not be fully satisfied.

Q-Q Plot (Q-Q Plot):

The Q-Q Plot shows that the residuals deviate from the theoretical normal distribution in the tails (the upper and lower points deviate from the reference line), which indicates that the residuals may not be normally distributed and that there may be a heavy-tailed distribution or outliers.

Non-normal residuals may affect the statistical inference results of the model, so further examination of outliers in the data or consideration of transformations of the variables is required.

Box-Cox Transformation:

The Box-Cox plot shows the optimal transformation parameter λ=1/4, indicating that the response variable may need to be log-transformed to improve the fit of the model.

Therefore, we should add second and third-order terms into the model and apply log transformation to y. Also, as the dummy of sex has a high p-value(0.768) of male, which means there is no statistically significant difference between female and male, we set Female and Male as the same category.

2.2 transform Y and delete variables

```
data$Sex[data$Sex == 'M'] = 'F'
test = data[ind, ]
train = data[-ind, ]

x1=train[,1]
x2=train[,2]
x3=train[,3]
x4=train[,4]
x5=train[,5]
x6=train[,6]
x7=train[,7]
x8=train[,8]
y=train[,9]
```

```
levels(train$Sex)
```

```
## [1] "F" "I" "M"
```

```
table(train$Sex)
```

```
##
##    F    I    M
## 1900  884    0
```

```
train = train[train$Sex != 'M', ]
test = test[test$Sex != 'M', ]
```

```
m1_tran = lm(sqrt(1/sqrt(y))  ~ Sex + Diameter + Height + Whole_weight + Shucked_weight + Shell_weight, data = tr
ain)
summary(m1_tran)
```

```
##
## Call:
## lm(formula = sqrt(1/sqrt(y)) ~ Sex + Diameter + Height + Whole_weight +
##     Shucked_weight + Shell_weight, data = train)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.093294 -0.013257  0.002123  0.015445  0.142766
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)     0.655783   0.003594 182.482  < 2e-16 ***
## SexI            0.010666   0.001185   8.998  < 2e-16 ***
## Diameter       -0.200049   0.013936 -14.355  < 2e-16 ***
## Height         -0.292437   0.030064  -9.727  < 2e-16 ***
## Whole_weight   -0.034481   0.007320  -4.711 2.59e-06 ***
## Shucked_weight  0.167526   0.010125  16.545  < 2e-16 ***
## Shell_weight   -0.077152   0.013835  -5.577 2.69e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.02386 on 2777 degrees of freedom
## Multiple R-squared:  0.5936, Adjusted R-squared:  0.5927
## F-statistic:   676 on 6 and 2777 DF,  p-value: < 2.2e-16
```

```
vif(m1_tran)
```

```
##            Sex       Diameter         Height   Whole_weight Shucked_weight
##       1.489440       9.357360       6.567856      63.474575      24.791071
##   Shell_weight
##      18.150143
```

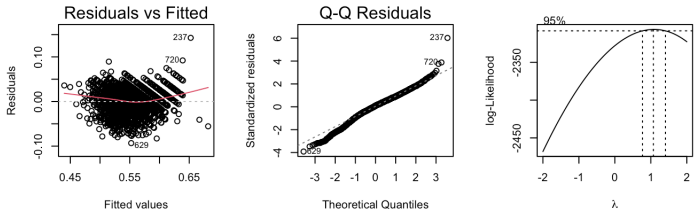
```
m2_tran = lm(sqrt(1/sqrt(y))  ~ Sex + Diameter + Height + Shucked_weight + Shell_weight, data = train)
summary(m2_tran)
```

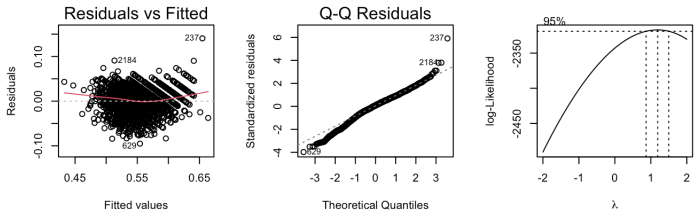
```
##
## Call:
## lm(formula = sqrt(1/sqrt(y)) ~ Sex + Diameter + Height + Shucked_weight +
##     Shell_weight, data = train)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.095113 -0.013077  0.001988  0.015549  0.140605
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)     0.657870   0.003580 183.771   <2e-16 ***
## SexI            0.011187   0.001185   9.442   <2e-16 ***
## Diameter       -0.204645   0.013955 -14.665   <2e-16 ***
## Height         -0.307095   0.030016 -10.231   <2e-16 ***
## Shucked_weight  0.125869   0.004950  25.427   <2e-16 ***
## Shell_weight   -0.127299   0.008870 -14.352   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.02395 on 2778 degrees of freedom
## Multiple R-squared:  0.5904, Adjusted R-squared:  0.5896
## F-statistic: 800.7 on 5 and 2778 DF,  p-value: < 2.2e-16
```

```
vif(m2_tran)
```

```
##            Sex       Diameter         Height Shucked_weight   Shell_weight
##       1.476489       9.311502       6.497491       5.880599       7.403923
```

```
par(mfrow=c(2,3))
plot(m1_tran, which=1)
plot(m1_tran, which=2)
boxcox(m1_tran)
plot(m2_tran, which=1)
plot(m2_tran, which=2)
boxcox(m2_tran)
```





The normality hypothesis is satisfied now.

2.3 Adding high order terms

```
none_mod_2 = lm(sqrt(1/sqrt(y)) ~ 1, data=train) #model with only intercept
full_mod_2 = lm(sqrt(1/sqrt(y)) ~ Sex + Diameter + Height + Shucked_weight + Shell_weight
                + I(Diameter^2) + I(Height^2) + I(Shucked_weight^2) + I(Shell_weight^2)
                + I(Diameter^3) + I(Height^3) + I(Shucked_weight^3) + I(Shell_weight^3),
                data=train)
#summary(full_mod_2)

#forward stepwise based on AIC
m3_tran=stepAIC(none_mod, scope=list(upper=full_mod_2, lower = ~1), direction="both", k=2, trace = FALSE)
vif(m3_tran)
```

```
##        Shell_weight I(Shucked_weight^2)              Height                 Sex
##           140.891359           21.768568           71.135561            1.535245
##   I(Shell_weight^2)      Shucked_weight   I(Shell_weight^3)         I(Height^2)
##           309.097756           37.377433           72.254563           61.949630
```

```
summary(m3_tran)
```

```
##
## Call:
## lm(formula = y ~ Shell_weight + I(Shucked_weight^2) + Height +
##     Sex + I(Shell_weight^2) + Shucked_weight + I(Shell_weight^3) +
##     I(Height^2), data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.1510 -1.3307 -0.3605  0.8704 16.6870
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)           4.7641     0.4806   9.912  < 2e-16 ***
## Shell_weight         47.5748     3.5641  13.348  < 2e-16 ***
## I(Shucked_weight^2)   7.6457     0.9292   8.228 2.89e-16 ***
## Height               43.0928     9.1484   4.710 2.59e-06 ***
## SexI                 -0.7152     0.1113  -6.427 1.52e-10 ***
## I(Shell_weight^2)   -55.4882     8.7447  -6.345 2.58e-10 ***
## Shucked_weight      -18.8966     1.1495 -16.438  < 2e-16 ***
## I(Shell_weight^3)    31.0144     6.6605   4.656 3.37e-06 ***
## I(Height^2)         -81.8007    31.3166  -2.612  0.00905 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.206 on 2775 degrees of freedom
## Multiple R-squared:  0.5284, Adjusted R-squared:  0.5271
## F-statistic: 388.7 on 8 and 2775 DF,  p-value: < 2.2e-16
```

```
#forward stepwise based on BIC
m4_tran=stepAIC(none_mod, scope=list(upper=full_mod_2, lower = ~1), direction="both", k=log(n), trace = FALSE)
vif(m4_tran)
```

```
##        Shell_weight I(Shucked_weight^2)              Height                 Sex
##           108.477441           21.589272            6.698631            1.535245
##   I(Shell_weight^2)      Shucked_weight   I(Shell_weight^3)
##           218.242309           37.274960           54.596114
```

```
summary(m4_tran)
```

```
##
## Call:
## lm(formula = y ~ Shell_weight + I(Shucked_weight^2) + Height +
##     Sex + I(Shell_weight^2) + Shucked_weight + I(Shell_weight^3),
##     data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.1015 -1.3195 -0.3769  0.8591 16.7739
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)           5.8351     0.2511  23.241  < 2e-16 ***
## Shell_weight         52.0402     3.1306  16.623  < 2e-16 ***
## I(Shucked_weight^2)   7.4254     0.9264   8.016 1.60e-15 ***
## Height               20.3496     2.8103   7.241 5.73e-13 ***
## SexI                 -0.7154     0.1114  -6.422 1.57e-10 ***
## I(Shell_weight^2)   -67.8720     7.3556  -9.227  < 2e-16 ***
## Shucked_weight      -18.7394     1.1492 -16.307  < 2e-16 ***
## I(Shell_weight^3)    39.6151     5.7958   6.835 1.00e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.208 on 2776 degrees of freedom
## Multiple R-squared:  0.5273, Adjusted R-squared:  0.5261
## F-statistic: 442.3 on 7 and 2776 DF,  p-value: < 2.2e-16
```

3. Model prediction

```
# predict test set
pred1 <- predict(m1_tran, newdata = test)
pred2 <- predict(m2_tran, newdata = test)
pred3 <- predict(m3_tran, newdata = test)
pred4 <- predict(m4_tran, newdata = test)

# actual value
actual <- test$y
```

```
evaluate_model <- function(pred, actual) {
  mse <- mean((pred - actual)^2)
  rmse <- sqrt(mse)
  mae <- mean(abs(pred - actual))
  r2 <- 1 - sum((pred - actual)^2) / sum((actual - mean(actual))^2)
  return(c(MSE = mse, RMSE = rmse, MAE = mae, R2 = r2))
}

results <- data.frame(
  Model1 = evaluate_model(pred1, actual),
  Model2 = evaluate_model(pred2, actual),
  Model3 = evaluate_model(pred3, actual),
  Model4 = evaluate_model(pred4, actual)
)
print(results)
```

```
##         Model1    Model2    Model3    Model4
## MSE  129.84863 129.85050 4.5444508 4.5531474
## RMSE  11.39511  11.39520 2.1317717 2.1338105
## MAE   10.91288  10.91297 1.5493376 1.5497591
## R2   -11.22876 -11.22893 0.5720171 0.5711981
```

From the test result table:

MSE: Model 3 (4.5444508) < Model 4 (4.5531474) < Model 1 (129.84863) < Model 2 (129.85050) RMSE: Model 3 (2.1317717) < Model 4 (2.1338105) < Model 1 (11.39511) < Model 2 (11.39520) MAE: Model 3 (1.5493376) < Model 4 (1.5497591) < Model 1 (10.91288) < Model 2 (10.91297) R^2: Model 3 (0.5720171) > Model 4 (0.5711981) > Model 1 (-11.22876) > Model 2 (-11.22893)

As model 3 has the smallest MSE, RMSE, MAE, and larger R^2 on the test set, we can say it is the best predicting model.

```
pred1_train <- predict(m1_tran, newdata = train)
pred2_train <- predict(m2_tran, newdata = train)
pred3_train <- predict(m3_tran, newdata = train)
pred4_train <- predict(m4_tran, newdata = train)

actual_train <- train$y
```

```
model_overfit <- function(pred_train, pred_test, actual_train, actual_test) {
  mse_train <- mean((pred_train - actual_train)^2)
  mse_test <- mean((pred_test - actual_test)^2)
  return(c(Train_MSE = mse_train, Test_MSE = mse_test))
}

results_overfit <- data.frame(
  Model1 = model_overfit(pred1_train, pred1, actual_train, actual),
  Model2 = model_overfit(pred2_train, pred2, actual_train, actual),
  Model3 = model_overfit(pred3_train, pred3, actual_train, actual),
  Model4 = model_overfit(pred4_train, pred4, actual_train, actual)
)
print(results_overfit)
```

```
##            Model1   Model2   Model3   Model4
## Train_MSE 128.5656 128.5643 4.850329 4.862254
## Test_MSE  129.8486 129.8505 4.544451 4.553147
```

As all of the models have similar MSE for training sets and test sets, they are not overfitted.